Reading. Read the following sections in the textbook.

- Ch 3: 3.6 through 3.6.7

Homework Questions. Do the following problems and turn in your answers as a single PDF document to blackboard. For each of the programming problems below, you must turn in your source code along with output showing your program compiles, runs, and produces the correct output. Note that this should be done on multiple test input strings for each program. Your homework is due in blackboard at any time on the due date.

1. Do Practice Problem 3.17 from the textbook and write down your answers.

2. Do Practice Problem 3.18 from the textbook and write down your answers.

3. Write an assembly program that converts a lower-case string \texttt{str} to an upper case string and outputs the result, assuming the original string consists of all lower case letters. The basic “skeleton” of your program should be as follows. Be sure to test your program on different example strings.

```assembly
# ------------------------------------------------------------
# File: hw6_3.s
# Description ...
# ------------------------------------------------------------

.global _start
.text

_start: ... put our code here ...

# write to stdout
movq $1, %rax
movq $1, %rdi
movq $str, %rsi
movq $len, %rdx

# exit program
movq $60, %rax
xorq %rdi, %rdi
syscall

.data
str: .ascii "helloworld\n"   # any string here
.equ len, (. - str)
```
4. Write an assembly program that adds 1 character to each lower-case letter in the given string, wrapping around from z to a. As above, your program should output the resulting string. Assume that each character in the string is a lower-case letter. You should use the same basic program “skeleton” as above. Be sure to test your program using different strings.

5. Write an assembly program that flips the case of the letters, but leaves the rest of the characters unchanged. For instance, if a letter in the string is lower case, the result should be the same letter in upper case. Whereas, if a letter in the string is upper case, the result should be the same letter in lower case. As above, your program should output the resulting string. You should use the same basic program “skeleton” as above. Be sure to test your program using different strings. Note that your program should not use any conditional move instructions.

6. Rewrite your program in question 5 to use conditional move instructions. Note that the smallest portion of a register that can be moved is a word. Therefore, when using cmov you will need to move a full word as opposed to just a byte. The general approach is to compute both the upper and lower of %bl and then instead of jumping to do the conversion, use a conditional move. The general structure of your checks should be to first see if %bl is less than A or greater than z (in which case, you should just write out the character and skip the conditional moves), then conditionally move if it is less than or equal to Z or greater than or equal to a, and then write out the character (which has either been converted to lower case, upper case, or left unchanged).

7. Write an assembly program similar to the above programs that finds the largest lower-case letter in a string and outputs the letter. Assume the input string consists of one or more lower-case letters, and only consists of lower-case letters. Note that you will need to modify the part of the “skeleton” code that writes the string to standard output. Test your program using different strings.

8. Write an assembly program similar to the above programs that “flips” two words in a string by “rotating” the letters. Your program should output the flipped words. Assume the given string consists of two words separated by a space, where each word has at least one character. As an example, if the string is "foo bar\n", your program should successively rotate the string until it ends up as "bar foo\n". For this example, your program should first modify the string to be " foobar\n". Then, the complete flipping of words requires three rotations:

- 1st Rotation: "r fooba\n"
- 2nd Rotation: "ar foob\n"
- 3rd Rotation: "bar foo\n"

Again, be sure to test your program over different strings.